
INSTALL ME MAYBE

Turning claimable VS Code extension IDs into
supply-chain attacks



INTRODUCTION

\$ whoami

- Security Researcher @ Aikido Security
- Focus on malware and SCS vulnerability research
- Like to hack sh!t on my free time



@0x_rcss



aikido.dev



raphaelcssilva

INTRODUCTION

\$ whoami

- Security Researcher @ Aikido Security
- Focus on malware and SCS vulnerability research
- Like to hack sh!t on my free time
- Horrible at making pretty slides (sorry)



LAST YEAR...

PLUGINS GONE ROGUE

Attacking Developer Environments



APPSEC VILLAGE

Raphael Silva
Security Researcher @ Checkmarx

LAST YEAR...

Malicious VSCode extensions with millions of installs discovered

By **Bill Toulas**

 June 9, 2024  10:22 AM  7

RL Blog

Threat Research | April 3, 2024

Malicious helpers: VS Code Extensions observed stealing sensitive information

SECURING THE CLOUD MAY 16, 2023

VSCode Security:
Malicious Extensions
Detected- More Than
45,000 Downloads-
PII Exposed, and
Backdoors Enabled



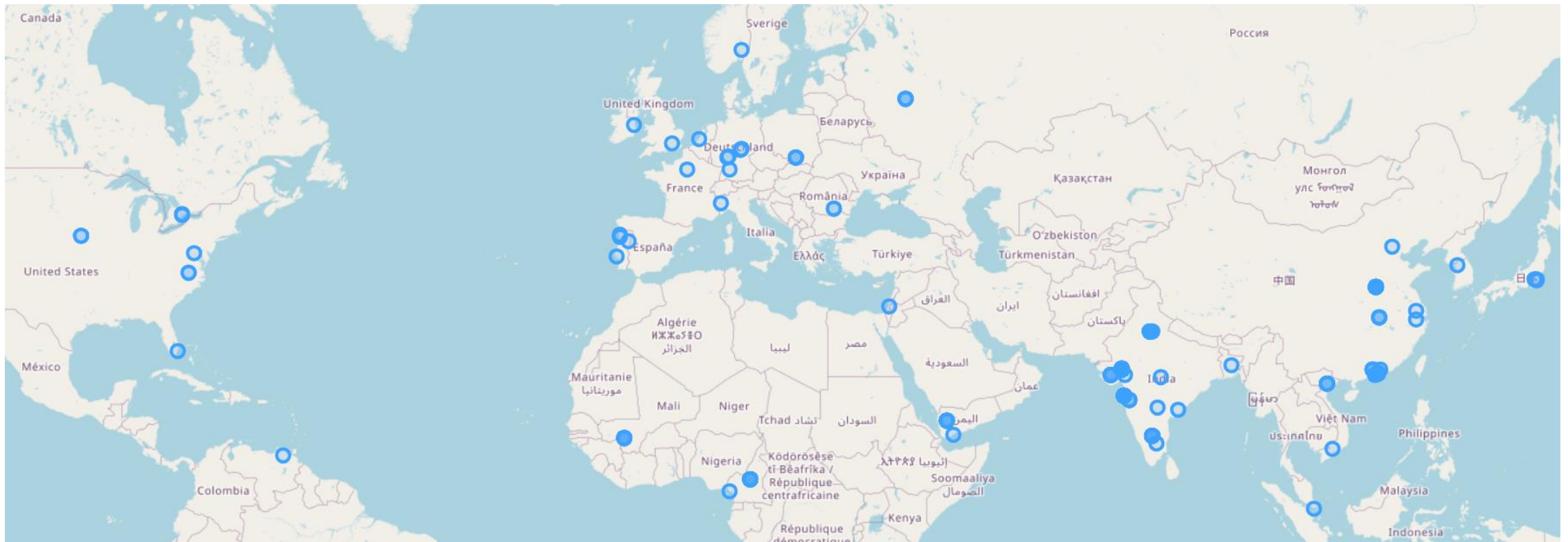
By Ori Abramovsky, Head Of Data Science,
Cloud Security

VSCode extensions found downloading early-stage ransomware

By **Bill Toulas**

 March 20, 2025  03:54 PM  0

LAST YEAR...



WHY DEVELOPER ENVIRONMENTS MATTER



Developers have high-privilege system access



IDEs contain sensitive credentials and source code



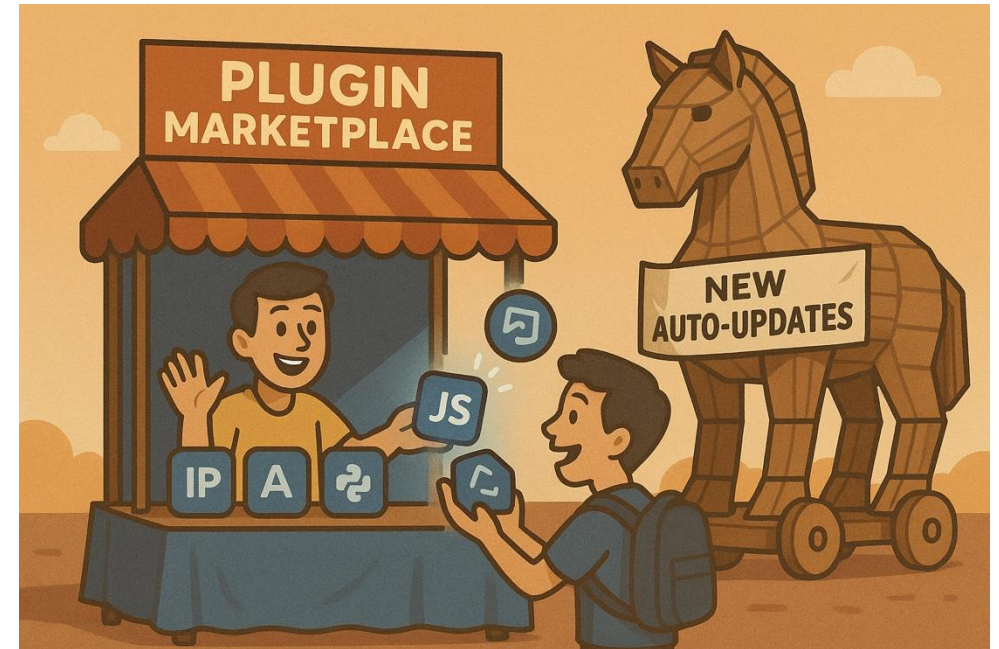
Trusted development tools bypass security controls



Single compromised plugin can impact ecosystem

WHY VS CODE IS AN ATTACKER'S DREAM

- Massive reach (in ~75 % of developer desktops)
- Anyone can publish
- Auto-updates
- Trust signals are flimsy
- Full IDE privileges
- Forks like **Cursor**, **Windsurf**, **Antigravity**, etc. already make up 18–22% of market share.



VS CODE EXTENSION ANATOMY

- Javascript Project
- Packaged as .vsix. A glorified ZIP that VS Code unpacks into the user profile
- Manifest-driven – package.json (name, publisher, version, scripts, dependencies, etc)
- Dependencies
- activationEvents (when does the extension run?)
- extensionPack, extensionDependencies

PACKAGE.JSON

```
{  
  "publisher": "ms-python",  
  "name": "python",  
  "activationEvents": [  
    "onStartupFinished"  
  ],  
  "extensionDependencies": [ "..." ]  
}
```

SILENT TRIGGERS: ACTIVATIONEVENTS & SCRIPTS

- onStartupFinished
- onLanguage
- * – fires at every IDE launch
- ...

```
],  
"activationEvents": [  
  "*",  
  ~~~~~  
],
```

```
[info] ExtensionService#_doActivateExtension Expressjs.expressjs-session, startup: false, activationEvent: "*"
```

DEPENDENCIES (SCS 101)

- NPM
- GitHub
- URL
- Local
- ...

```
},  
"dependencies": {  
  "express": "https://expressjs.com/",  
}
```

```
},  
"dependencies": {  
  "express": "^4.18.2",  
  "expressjs-session": "^4.4.0"  
},  
"devDependencies": {  
  "@types/vscode": "^1.100.0",  
  "@types/mocha": "^10.0.10",  
  "@types/node": "20.x",  
  "eslint": "^9.25.1",  
  "@vscode/test-cli": "^0.0.10",  
  "@vscode/test-electron": "^2.5.2"  
}
```

```
{  
  "name": "foo",  
  "version": "0.0.0",  
  "dependencies": {  
    "express": "expressjs/express",  
    "mocha": "mochajs/mocha#4727d357ea",  
    "module": "user/repo#feature\branch"  
  }  
}
```

AN EXTENSION IS A LONG-LIVED PROCESS INSIDE YOUR EDITOR.

SEES

Workspace files. Open buffers. Project tree.

TOUCHES

Terminals. Tasks. Debug sessions. Git.

READS

Env vars. `~/ .aws`. `~/ .ssh`.
Local files.

RUNS IN

Local. WSL. SSH remote.
Devcontainer.

Secrets, source code, etc...

THE MAGIC STRING

```
h > {} package.json > ...  
{  
  "name": "python",  
  "displayName": "Python",  
  "description": "Python lang",  
  "version": "2026.4.0",  
  "featureFlags": { ...  
},  
  "capabilities": { ...  
},  
  "publisher": "ms-python",
```

More Info

Version	2026.5.2026070801
Released on	1/19/2016, 7:03:11 AM
Last updated	7/8/2026, 4:35:23 AM
Publisher	Microsoft
Unique Identifier	ms-python.python
Report	Report a concern

WHAT IS PUBLISHER.EXTENSION REALLY?

PUBLISHER

ms-python

A namespace on a marketplace.
Owned by an account. Marketplace-specific.

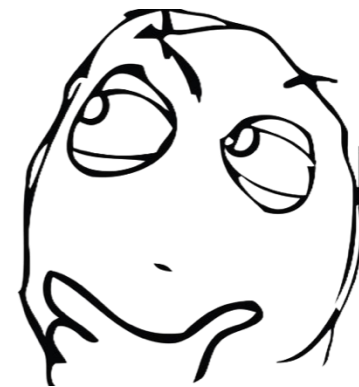
EXTENSION NAME

python

The actual package, declared in `package.json`.
A different person can publish the same name elsewhere.

THE CATCH

Looks portable but the **identity is marketplace-specific**.



SAME STRING. DIFFERENT REGISTRY.

REGISTRY A

VS Code Marketplace

Microsoft-operated.
Used by stock VS Code.
~70k extensions.

`publisher.extension` — OWNED

REGISTRY B

Open VSX

Eclipse Foundation.
Used by VS Code-derived editors.
Different namespace ledger.

`publisher` — UNCLAIMED

≠

SAME STRING. DIFFERENT REGISTRY.

Project Details

 [Microsoft/vscode-python](https://github.com/Microsoft/vscode-python)

More Info

Version	2026.5.2026070801
Released on	1/19/2016, 7:03:11 AM
Last updated	7/8/2026, 4:35:23 AM
Publisher	Microsoft
Unique Identifier	ms-python.python
Report	Report a concern

Version

2026.4.0 ▼

Published 5 months ago

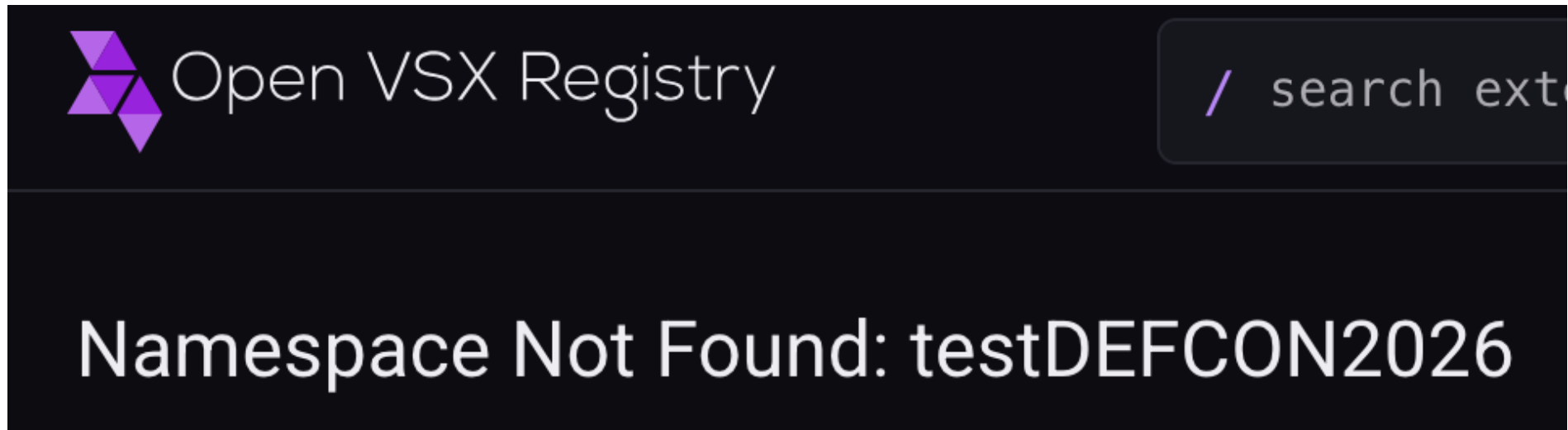
Version Alias

latest

Unique Identifier

ms-python.python

SAME STRING. DIFFERENT REGISTRY.



PUBLISHING EXTENSIONS

- Create a GH account
- Create an Eclipse Account
- Link GH to Eclipse
- Create Open VSX account
- Connect with Eclipse
- Done!



PREVIOUS WORK

KOI RESEARCH

How We Prevented Cursor, Windsurf & Google Antigravity from Recommending Malware



Oren Yomtov

January 6, 2026



A FAMILIAR FACE?

2021 · ALEX BIRSAN

Dependency confusion in package managers.

Internal package name → resolved through public registry → attacker had published a higher version. Code execution at Apple, Microsoft, PayPal, dozens more.

2026 · THIS TALK

Dependency confusion for editor extensions.

Trusted extension ID → resolved through a registry that doesn't have it → attacker claims the namespace. Code execution in developer environments.

Same primitive, aimed somewhere new. This time what gets confused is a publisher namespace.

NAME-TAKEOVER, MEET DEPENDENCY CONFUSION.

FAMILY A

Name takeover

Trusted reference → unowned namespace.
Examples: dangling GitHub Apps, GitHub RepoJacking.

FAMILY B

Dependency confusion

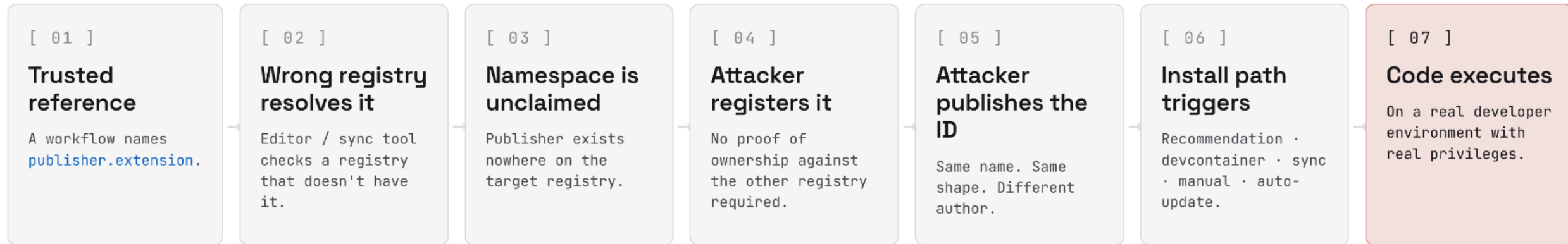
Same name, multiple registries. The tool picks the wrong one. Birsan 2021, and every variant since (e.g.: npx confusion).

INTERSECTION

Extension Confusion

The trusted name *and* the claimable namespace meet in editor tooling.

THE GENERAL ATTACK MODEL



WHERE IDS TRAVEL 1/5

`.vscode/extensions.json`

`.VSCODE/EXTENSIONS.JSON`

```
{
  "recommendations": [
    "esbenp.prettier-vscode",
    "dbaeumer.vscode-eslint",
    "internal-team.lint-helper"
  ]
}
```

Lowest (but still some) friction
install path in the editor.

- Opening the repo prompts to install.
- Lives in the repo — forks, merges, gets copy-pasted.
- Rarely audited like a dependency file.
- Lives in `.code-workspace` and team templates too.

WHERE IDS TRAVEL 2/5

.devcontainer/devcontainer.json

DEVCONTAINER.JSON

```
{
  "image": "mcr.microsoft.com/devcontainers/python:3",
  "customizations": {
    "vscode": {
      "extensions": [
        "ms-python.python",
        "acme-corp.deploy-tools"
      ]
    }
  }
}
```

Open repo. Start container. You're done.

- Extension install becomes part of **trusted bootstrap**.
- Container resolution may differ from local.
- Often runs unattended — codespaces, remote dev hosts.

WHERE IDS TRAVEL 3/5

Setup scripts & onboarding docs.

SCRIPTS/BOOTSTRAP.SH

```
#!/usr/bin/env bash
# Set up dev env for new joiners

EXTS=(
  ms-vscode.cpptools
  redhat.vscode-yaml
  internal.platform-helper # written 2021, no one remembers why
)

for e in "${EXTS[@]}; do
  code --install-extension "$e" --force
done
```

The kind of file nobody reviews.

- Bootstrap scripts. README.md install lists.
- Devbox / Nix / Docker base images.
- Wiki onboarding pages. Slack pins.
- Internal “recommended extensions” sheets that nobody updates.

Stale references age into install paths.

WHERE IDS TRAVEL 4/5

IDE migration. Settings sync. Forks.

EDITOR MIGRATION / IMPORT

“Bring my extensions from VS Code.” The source list assumes Microsoft’s registry. The destination may not.

THIRD-PARTY SYNC TOOLS

Community scripts and extensions.

SPECIFIC NOTE

Cursor looked more defensive in the missing-extension import flow I tested. The broader marketplace fragmentation issue still applies across other editors, sync tools, and devcontainer paths.

WHERE IDS TRAVEL 5/5

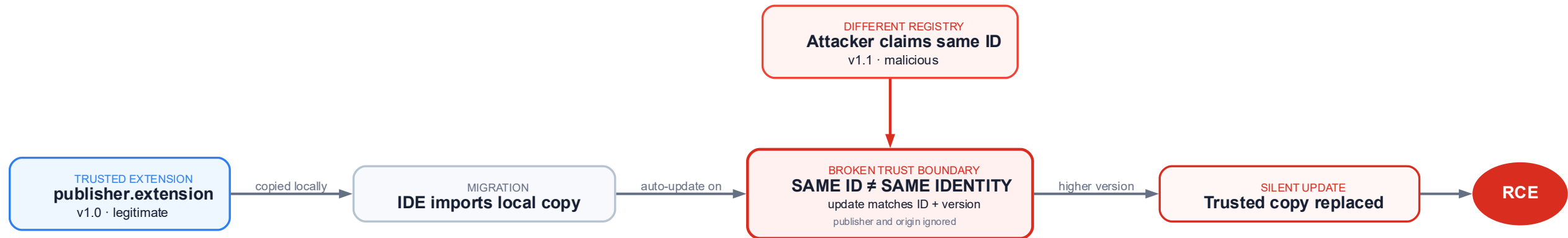
And then auto-updates make it worse.

- A claimable ID gets imported / synced / installed once.
- Attacker registers the namespace later.
- Attacker publishes v1.0.0 → v1.0.1.
- The editor pulls the higher version. **Silently.**
- Remove from marketplace later? Already installed copies stay.

```
// What the editor sees  
"publisher.extension": "^1.0.0"  
// What gets installed today  
attacker → "1.0.1" ✓ auto-update
```

Removal ≠ remediation. The installed copy keeps executing.

EXTENSION CONFUSION RECAP



FINDING CLAIMABLE IDS IN THE WILD

Scan, check, score. Twice.

PHASE 01 • DISCOVERY

- Public repo & setup-file sweeps for `publisher.extension` references.
- Existence check against the registry the target actually resolves through.
- Namespace claimability check (publisher present? owner active? ownership cross-verified?).
- Prioritize by prevalence, install counts, workflow context, org relevance.

PHASE 02 • COMPARISON

- Top-N VS Marketplace extensions × Open VSX availability.
- Top 5k → narrow but high-signal slice.
- Top 20k → the part that got uncomfortable.

RESULTS FROM SCANNING

Not just the long tail.

TOP 5,000

2,056

CLAIMABLE COMBOS

1,914

UNIQUE NAMESPACES

~187M

VS MARKETPLACE INSTALLS TIED TO VULN ENTRIES

TOP 20,000

12,660

CLAIMABLE COMBOS

11,013

UNIQUE NAMESPACES

~228M

VS MARKETPLACE INSTALLS TIED TO VULN ENTRIES

ENM - ETHICAL NON-MALWARE

Strict ethics. Minimum metadata.

COLLECTED – ATTRIBUTION ONLY

- ☒ Timestamp, IP, forwarded-for
- ☒ User agent, editor name + version
- ☒ Extension install path
- ☒ Remote context flag (local / WSL / SSH / devcontainer)
- ☒ Git email domain, git remote (when present)
- ☒ Host / user clues when available

NEVER COLLECTED

- ☐ Source files, code, or repo content
- ☐ Tokens, credentials, secrets
- ☐ Shell command output
- ☐ Environment variable values
- ☐ Persistence, post-exploitation, lateral movement
- ☐ Anything beyond proving execution

WHAT DID THE CALLBACKS LOOK LIKE?

CALLBACK.JSON (EXAMPLE)

```
{
  "ts":      "2026-03-14T09:42:17Z",
  "ip":      "203.0.113.47",
  "editor":  "vscode-derived/1.8.2",
  "ext_path": "/home/jdoe/.vscode-oss/extensions/publisher.extension-1.0.0",
  "remote":  "wsl",
  "git_email": "jdoe@example-org.com",
  "git_remote": "github.com/example-org/example-repo",
  "user":    "jdoe",
  "host":    "example-laptop-04"
}
```

SIGNALS THAT MATTERED

- Corporate git email domain
- Corporate git remote
- Repeated host / user pairs
- Editor metadata (variant + version)
- Remote-context flag (WSL / SSH / devcontainer)

IP alone was noisy. Stack the signals.

END RESULTS

• SINCE MARCH

1,818,216

callbacks from real developer environments.

END RESULTS BY THE NUMBERS

By the numbers.

TOTAL CALLBACKS

1,818,216

From benign PoC extensions under claimable IDs.

DISTINCT IPS

176,698

20,337 distinct hosts.

COUNTRIES

160+

Pretty much everywhere

REPORTS / DISCLOSURES

300+

BBPs, VDPs & direct contacts.

BOUNTIES SO FAR

\$270k+

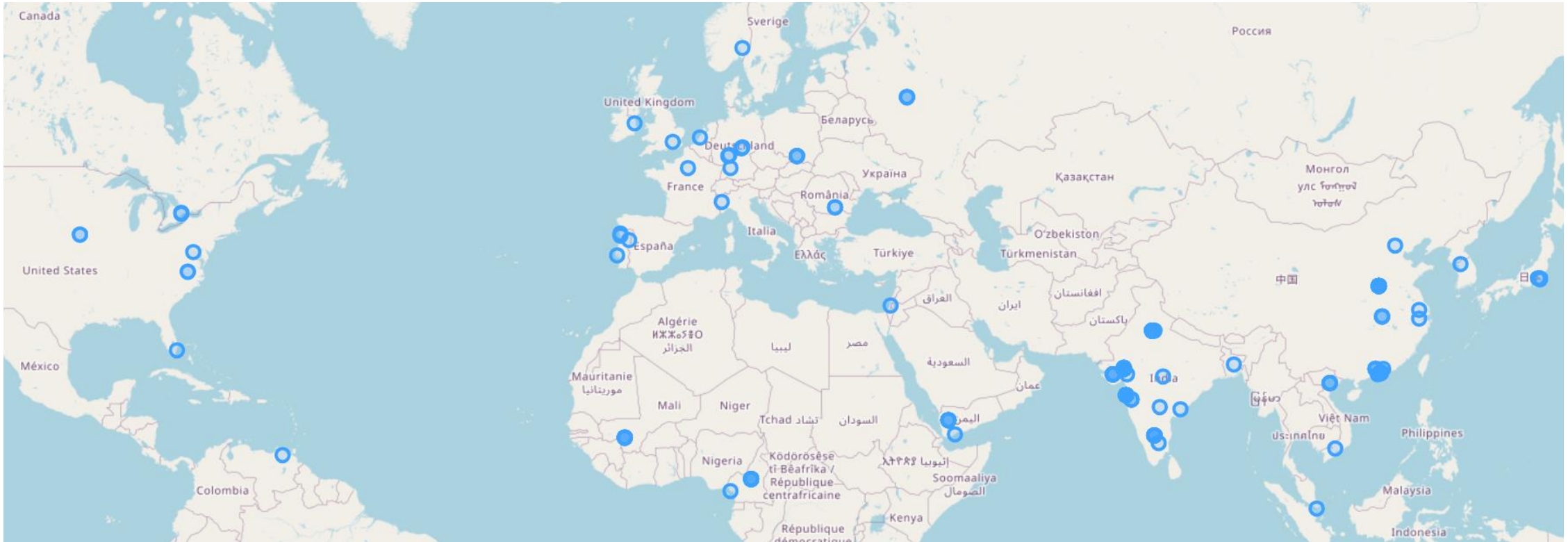
Multiple programs. Still climbing.

TOP-20K CLAIMABLE COMBOS

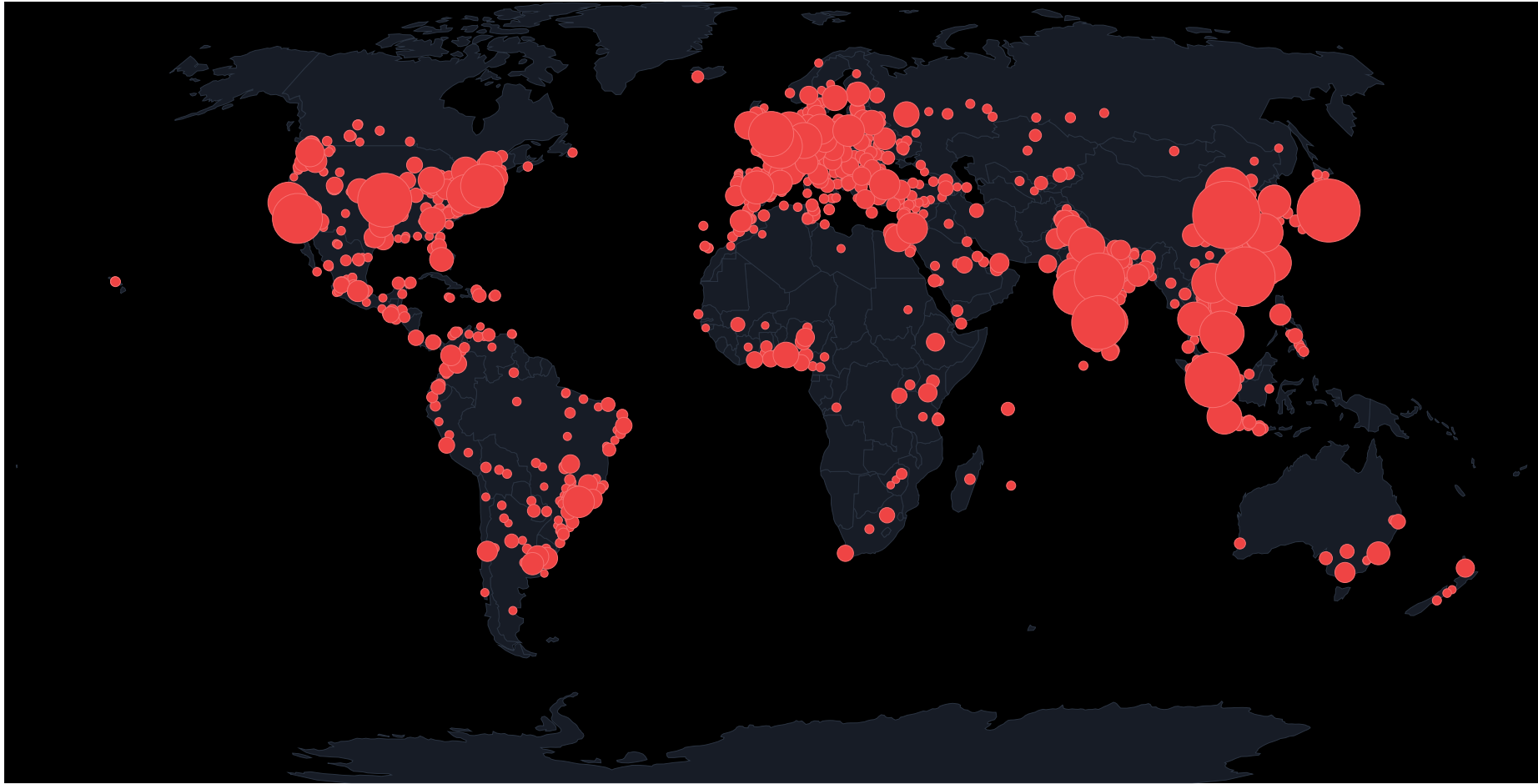
12,660

~420M VS Marketplace installs tied to them.

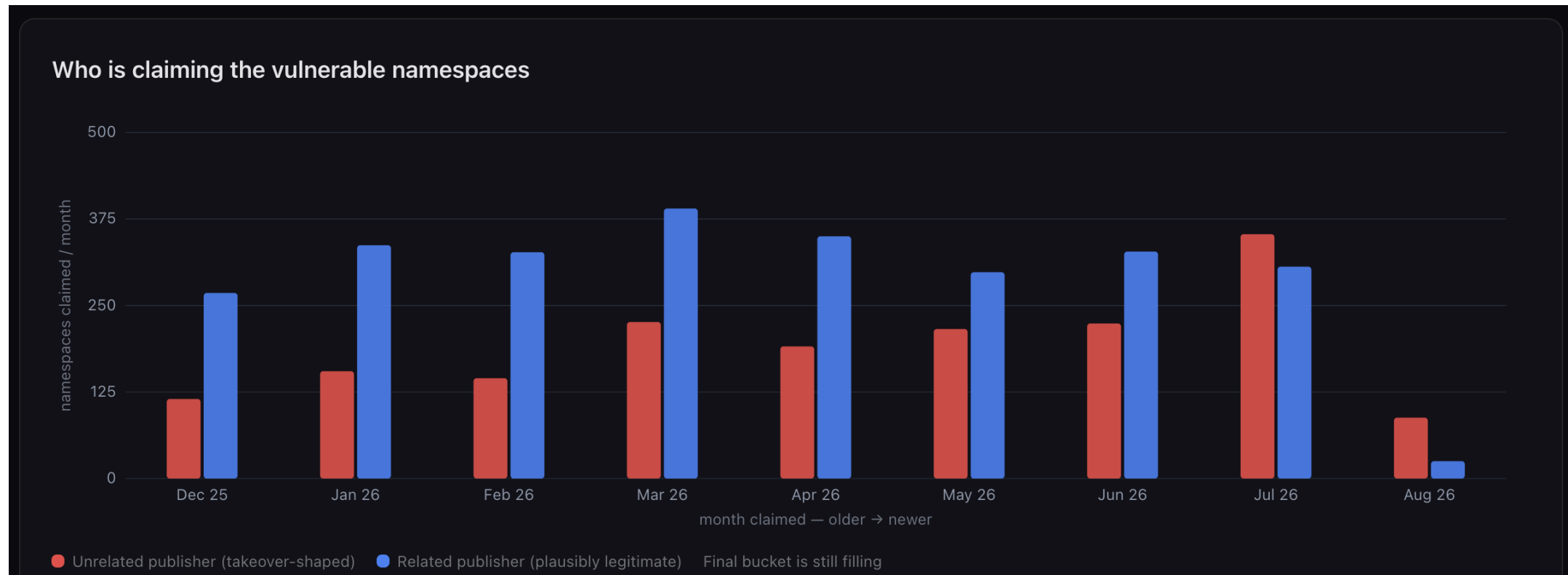
LAST YEAR...



4 MONTHS LATER



PEOPLE ARE CATCHING UP



DISCLOSURE PROCESS - REPORTS

- Sensitive nature of the whole thing
- Sheer number of cases to go through
- Novelty reports to most companies → Skepticism

You also have to deal with...

- Slow triage (influx of slop reports)
- Disputes (LOTS of re-explaining the issue)



DISCLOSURE PROCESS – WHAT DID THE REPORTS LOOK LIKE?

- Intro to who I am and what I am researching (with explanation of the issue in depth)
- What I observed/PoC (what triggered it, callbacks, important data points and attribution)
- Why I think this matters
- Different ways this can happen
- What I'm sharing
- Impact
- Remediation
- FAQ
- Attachments (example of an extension + all relevant callbacks)

DISCLOSURE PROCESS - RESPONSES

 [redacted] updated the severity from Critical (9.6) to None. [redacted]

Hi rcss,

Thank you — genuinely. This is one of the most thorough and well-constructed reports we've received, and we want to acknowledge both the quality of the write-up and the responsible way you carried out the research: benign test packages, metadata limited to what was needed for correlation, the callback URL disclosed up front, the PoC code shared, and the follow-up timeline detail. That care made it straightforward for us to understand and investigate, and we appreciate it.

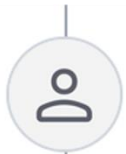
[redacted]
rcss, good morning! I come with even grater news. Your bounty has been released proportional to the P1 (Critical) severity of our Primary Section as the highest range of it, limiting the paycheck up to [redacted]. However, after a few internal alignments our team came to the conclusion of releasing additional [redacted] for you as a bonus due the relevance of your finding. :-) Thank you again for making of [redacted] safer place, and please keep the reports coming! Have a great week ahead. Best, [redacted]

[redacted] posted a comment. [redacted]

This is a great finding. If you publish, do not mention [redacted] per the program rules. This is a magical finding and teams are interested in it for both security reasons but also how it is like old-school phone freaking style hacking. Nice job.

[redacted] rcss with a [redacted]
We're paying out our maximum bounty range for your report. Thank you very much for this report. [redacted]

DISCLOSURE PROCESS – HOW NOT TO RESPOND



closed the report

This submission is intentionally outside of our program boundaries. The researcher conducted what amounts to a phishing awareness campaign targeting developers. This has nothing to do with finding vulnerabilities within our application. It also takes little-to-no effort, and instead, looks like a waterhole attack where he is capitalizing off of package confusion so he can then perform a shakedown of anyone who comes to the watering hole.

ETHICAL CONSIDERATIONS

- All extensions identified as research extensions
- Clear about what is being collected
- No harvesting of any excessive data
- Contact detail present on extensions
- Not able to receive commands after the fact
- Deletion of data after reports are done
- Responsible disclosure to all affected parties
- Drive changes to make community safer

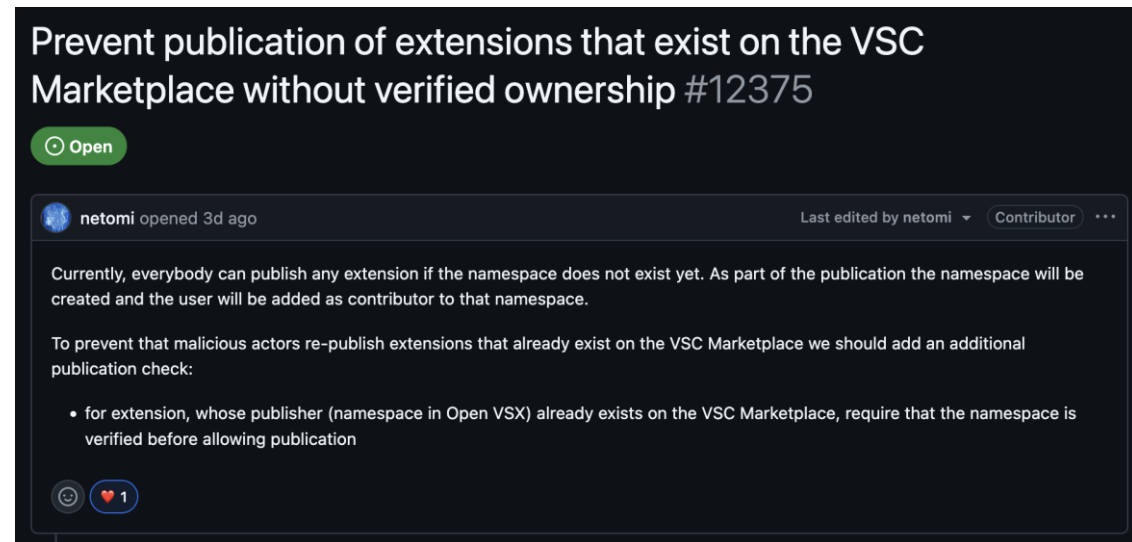


MITIGATIONS

- Implement an org-wide allowlist for approved editor extensions (per marketplace!)
- Block or restrict installation of unknown/unapproved extensions
- Disable automatic extension updates when possible
- Introduce a review/cooldown period before newly published extensions/versions are allowed to be installed, ideally 48h+
- Endpoint protection
- Review internal templates, devcontainers, bootstrap scripts, and onboarding docs so only approved extension identifiers are referenced

WHAT NOW? GOOD OR BAD FIRST?

- Prevent publication of extensions that exist on the VSC Marketplace without verified ownership (Issue #12375)
- Extension names will be required to be unique across Open VSX, including across different namespaces

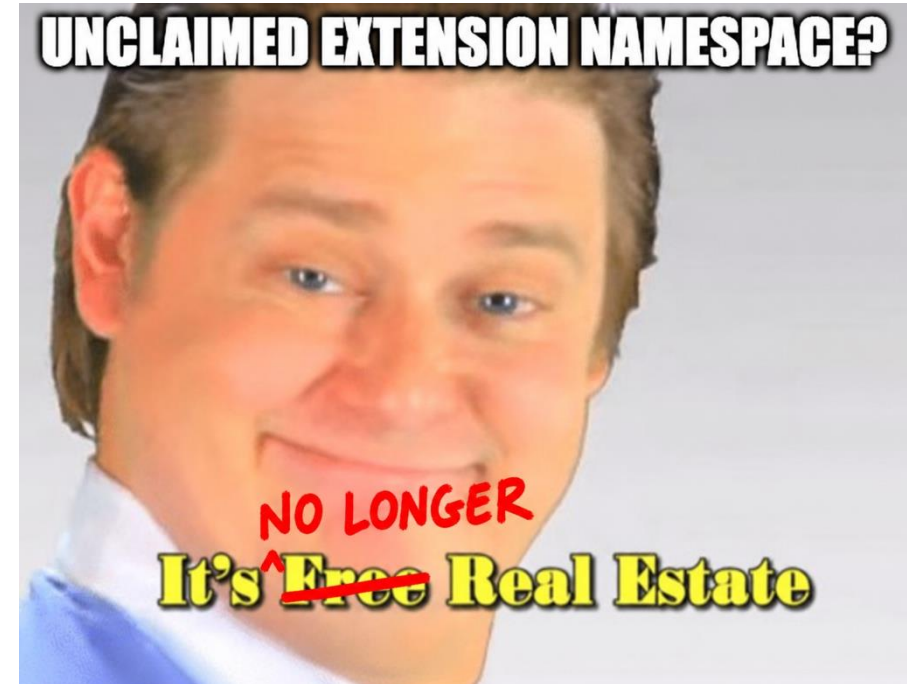


OPEN VSX OWNERSHIP CHECKS

- Checks publisher namespaces against VS Code Marketplace
- Requires verified ownership for matching namespaces
- Quarantines unverified matches pending ownership review

See:

- <https://github.com/EclipseFdn/open-vsx.org/issues/12375>
- <https://blogs.eclipse.org/post/brian-king/stronger-safety-nets-vs-code-marketplace-namespace-matching-open-vsx>



INSTALL ME MAYBE

Turning claimable VS Code extension IDs into
supply-chain attacks



@0x_rcss



raphaelcssilva

